INVENTION TITLE

System and Method to distribute reasoning and pattern matching in forward and backward chaining rule engines

DESCRIPTION

Background of the Invention

[Para 1] The RETE algorithm was invented by Dr. Charles Forgy in 1978 and re-published in 1982 titled "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", in Artificial Intelligence, 19 (1982) pp. 17-37. RETE was originally designed to improve the performance of forward chaining rule engines. The algorithm can be divided into two parts: the rule compiler and the runtime. The rule compiler analyzes a set of rules and creates a rooted acyclic graph. The conditions of the rule are converted to one and two input nodes. The action of the rule is converted to a terminal node. The second part of the algorithm is the runtime, which consists of a working memory and an agenda. Data requiring evaluation is added to the working memory and traverses the acyclic graph generated by the rule compiler. When all conditions of the rule are satisfied, the data triggering the rule are added to the agenda. RETE algorithm uses match-resolve-act cycles to evaluate the data. When all the rules have been fired, the evaluation process is complete. Each one input node in the acyclic graph remembers which data satisfies the condition in an alpha memory. Each two input node contains an alpha memory for the right side and a beta memory for the left side. The two input node uses the memory to remember which data matches the other side.

[Para 2] RETE uses the one input nodes to evaluate literal constraints, and two input nodes to evaluate joins. The rule compiler analyzes the rules for similarities and organizes the nodes to optimize performance and scalability. Forward chaining rule engines are said to be data driven. The rule engine can only reason over data that is added to the working memory.

Summary of the Invention

[Para 3] The invention solves performance issues previously ignored by researchers in the field of artificial intelligence. Using existing techniques, a rule engine implementing Dr. Forgy's original RETE algorithm cannot reason over extremely large datasets or partitioned data. The invention solves the following limitations.

- Dividing the pattern matching between multiple rule engines and share the partial and complete matches efficiently.
- Efficiently dividing the pattern matching across several rule engines dynamically.
- Efficiently distributing the indexes of the working memory across multiple machines.
- Efficiently distributing one or more conditional elements of a rule to remote systems and dividing the work across a cluster of computer systems.

[Para 4] In generally, one form of the invention is a computer system which implements the RETE extension to perform distributed reasoning. The system distributes reasoning by replicating the memory indexes across the systems. This reduces the memory requirements of each system and reduces the quantity of data each rule engine sends to the others. Using this approach, each rule engine in the cluster has a subset of the entire dataset and the complete indexes for the rules loaded in that engine. A system implementing the RETE extension can be used in a variety of applications like order management, regulatory compliance, military command control applications or business process automation. The invention is particularly well suited to large real-time systems like military command control systems or security trading systems, which partition data across several locations.

Detailed Description of the preferred embodiment

[Para 5] Distributed reasoning, unlike collaborative agents provides methods by which multiple rule engines reason over large datasets in real-time. Whereas collaborative agents use agents to reason over discrete problems, it cannot reason over large sets of data. Furthermore, collaborative agents" techniques require the rule engine to load all necessary data within the same engine. For small datasets, these techniques prove to be powerful, but they do not scale for large datasets.

[Para 6] A forward chaining rule engine utilizing distributed RETE algorithm, can reason over large datasets when nodes are distributed to other systems. This allows the system to match on a specific instance of a class (also called business object) without requiring the object instance reside in the same working memory and be locally available. It is important to note the engine will reason over shared data resident in other engines using data steams or indexes. This reduces the memory requirements and improves efficiency. It also enables the engines to share all or part of their reasoning network and distribute the process. Distributing nodes ~~of a set of rules~~ across multiple systems allows each engine to perform pattern matching on a single object instance and route the results to the originating system. Unlike load balancing techniques, a true distributed reasoning system does not require all systems of a cluster to deploy identical sets of rules, which creates redundant rules within the environment and increases maintenance costs. In a distributed reasoning/distributed pattern matching system, each system deploys a different set of rules (also called module or rule set), based on its own needs and configuration requirements. At runtime, the rule engines monitor resource utilization and distribute nodes dynamically and on demand. In some cases, it may be desirable to deploy the same set of rules across multiple instances. At runtime, the cluster would intelligently partition the data.

[Para 7] Prior techniques relied on load balancing techniques to prioritize and categorize atomic processes. This approach requires every system ~~to~~ have all required data locally, leading to multiple data caches. In a production environment with large datasets, each system cannot load all required data and data synchronization becomes a potential problem. Prior research into multi-threaded rule engines were unsuccessful because of the cost of maintaining locks. Distributed reasoning does not use locking mechanism and does not suffer from synchronization issues.

[Para 8] The invention described in the patent was previously impractical due to the processing power of personal computers. Researchers in the field of Artificial intelligence did not consider these techniques and have not considered them. The invention is currently feasible due to advances in the processing power of personal computers and high-speed networks. With this invention, it is possible to build large distributed reasoning systems using hundreds and thousands of personal computers. By distributing the indexes and facts between thousands of systems effectively, a forward and backward chaining rule engine can reason over hundreds of millions of facts using inexpensive hardware. Without this invention, the only technique for reasoning over large datasets is to get larger mainframes with more memory and processing power. As research has shown, scaling up the hardware rapidly reaches a limit and cannot scale to the same level as a distributed reasoning system.

Description of the Drawings

[Para 9] Fig. 1 is a network diagram of an example rule with four one input, one two input and one terminal node. The example rule contains one literal constraint for each object type and one join node between the two data elements.

[Para 10] Fig. 2 is a block diagram illustrating data partitioned across several rule engines. Each engine has the same rules, but may also contain other rules, which are not shared by other engines. A set of data is divided across several engines to improve scalabilty and performance, enabling parallel processing.

[Para 11] Fig. 3 is a block diagram, which shows 2 rule engines starting with different set of rules at startup. Some time later at T_n some nodes from one rule engine are distributed to the other rule engine. Optionally, both system can start with the same set of rules. In both cases, the data is partitioned as Fig. 2 illustrates.